Paper: GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. [5]

Jiaxin Liu

Group Reading

October 7, 2021

Overview

1. Introduction
   - Over-sampling and under-sampling
   - SMOTE
2. GraphSMOTE
   - Feature extractor and synthetic node generation
   - Edge generator
   - GNN classifier
   - Optimization Objective
3. Experiments

Introduction

- Class imbalance problem
  - Algorithm-level: cost sensitive learning.
  - Data-level: re-sample the original dataset such as SMOTE[1].
  - Hybrid approaches.

Introduction

- Class imbalance problem
  - Algorithm-level: cost sensitive learning.
  - Data-level: re-sample the original dataset such as SMOTE[1].
  - Hybrid approaches.
- Re-sample [3]:
  - Over-sampling: random and focused over-sampling for minority class.
  - Under-sampling: random and focused under-sampling for majority class.

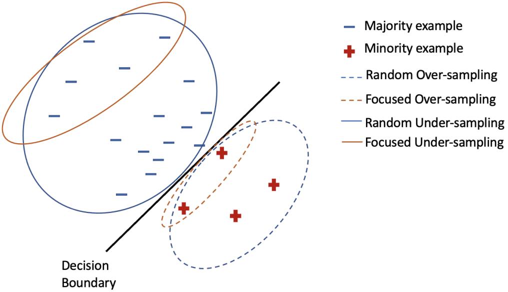


Figure: An example for re-sampling.

Synthetic Minority Over-sampling Technique (SMOTE)

- Over-sampling:
  - Replicate the original data.
  - Generate new synthetic data.
- SMOTE:
  - Over-sample the minority class.
  - Synthetic examples are introduced along the line segments joining any/all of the k minority class nearest neighbors.

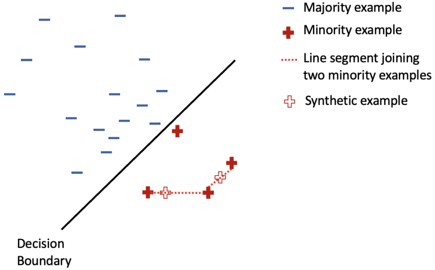


Figure: An example for SMOTE.

SMOTE

- Decision region for over-sampling the minority class with replication (left) and synthetic generation (right).

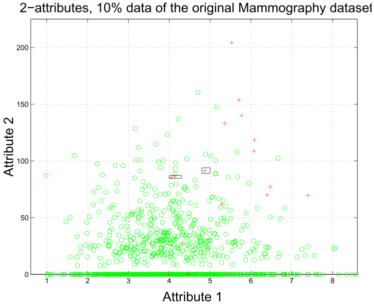


Figure: Decision region (solid line) as a result of replicating minority directly.

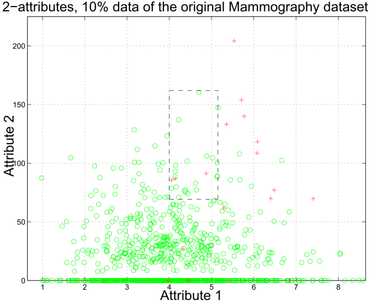


Figure: Decision region (dashed line) as a result of using synthetic data.

SMOTE and Mixup

Table: Comparison between SMOTE[1] and Mixup [4].

	SMOTE	Mixup
Source for the generation	Two minority examples	Any two examples
Class for the synthetic data	Minority class	$\lambda y_1 + (1-\lambda) y_2$
Model training	Original and synthetic data	Synthetic data only
Weakness	Know the neighbors' info	Inaccurate synthetic label

GraphSMOTE

Task: node classification task on graph $G = \{V, A, F\}$ in the transductive setting.

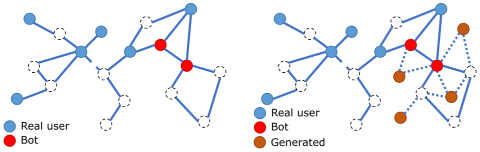
- $V = \{v_1, \cdots, v_n\}$ is a set of n nodes.
- $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix
- $F \in \mathbb{R}^{n \times d}$ denotes the node attribute matrix.
- $Y \in \mathbb{R}^{n}$ is the class information for node in G.
- V_L, Y_L denotes the nodes in the training set and their labels.
- m classes: $\{C_1, \cdots C_m\}$
- Imbalanced ratio: $\frac{\min_i |C_i|}{\max_i |C_i|}$ statisticized from V_L.

Goal: given the imbalanced node class set and a labeled training set V_L, find a node classifier $f(V, A, F) \to Y$ that works well for both majority and minority classes.

GraphSMOTE

Idea:

1. generate synthetic minority nodes → feature encoder and node generator;
2. assign links for these synthetic nodes → edge generator;
3. train the GNN on this augmented balanced graph → GNN classifier.



(a) Bot detection task (b) After over-sampling

Figure: An example of bot detection on a social network and the idea of over-sampling.

Feature extractor and synthetic node generation

- Feature extractor?
  - Raw node feature space is sparse and high-dimensional → hard to get similar nodes from the same class
  - Raw features don't consider the graph structure.
- Use one block of GraphSAGE [2] as the feature extractor.

$$h_v^1 = \sigma(W^1 \cdot \mathrm{CONCAT}(F[v,:], F \cdot A[:,v]))$$

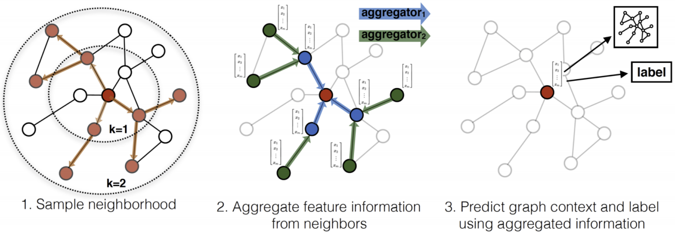


Figure: An example for GraphSAGE.

Node generation

Adopt SMOTE algorithm to generate synthetic node using the embedding features.

For a labeled minority node h_v^1 and ite label Y_v

1. Find closest labeled node to node h_v^1 in class Y_v.

$$nn(v) = \arg\min_u \|h_u^1 - h_v^1\|, \qquad Y_u = Y_v$$

2. Generate the synthetic node.

$$h_{v'}^1 = (1-\sigma)\cdot h_v^1 + \sigma \cdot h_{nn(v)}^1$$

where $\sigma \in [0,1]$, and $Y_{v'}^1 = Y_v$.

Edge generator

Edge generator models the existence of edges among nodes and can predict the edges for the synthetic nodes.

- Trained on the real nodes and existing edges.
- Used to predict the neighbor information for the synthetic nodes.

$$E_{v,u} = \text{softmax}(\sigma(h_v^1 \cdot S \cdot {h_u^1}^\top))$$

  where $E_{v,u}$ predicts the relation between node u, v and S is the parameter matrix.

- Loss function:

$$L_{edge} = \|E - A\|_F^2$$

- $\tilde{A}[v', u] = \begin{cases} 1, & \text{if } E_{v',u} > \eta \\ 0, & \text{otherwise} \end{cases}$ or $\tilde{A}[v', u] = E_{v',u}$.

GNN classifier

After adding the augmented nodes:

- Node representation: $H^1 \to \tilde{H}^1$
- Training set: $V_L \to \tilde{V}_L$
- Graph: $\tilde{G} = \{\tilde{A}, \tilde{H}\}$
  Introduce another block of GraphSAGE and a linear layer:

$$h_v^2 = \sigma(W^2 \cdot \text{CONCAT}(h_v^1, \tilde{H}^1 \cdot \tilde{A}[:, v])),$$

$$P_v = \text{softmax}(\sigma(W^c \cdot \text{CONCAT}(h_v^2, H^2 \cdot \tilde{A}[:, v])),$$

  where H^2 denotes the node representation from the second GraphSAGE block, W^2, W^c refer to the weight parameters.
- Loss

$$L_{node} = -\sum_{u \in \tilde{V}_L} \sum_{c} (1(Y_u == c) \cdot \log(P_v[c]))$$

Optimization Objective

Final objective function:

$$\min_{W^1, S, W^2, W^c} = L_{node} + \lambda \cdot L_{edge}$$

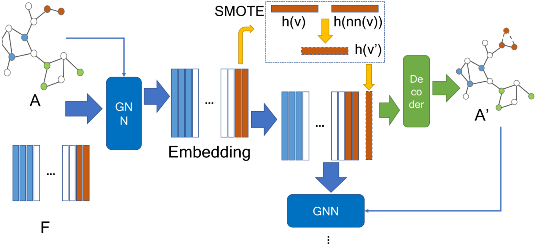


Figure: Overview of the framework.

Experiments

- Dataset:
  1. Cora: citation network containing 2708 papers from 7 areas.
     - Majority classes: each class have a training set containing 20 nodes.
     - Minority class: randomly samples 3 classes and down-sample $20 \times$ imbalance ratio(0.5)).
  2. BlogCatalog: $25\%, 25\%, 50\%$ for training, validation and test set.
     - Majority: .
     - Minority: 14 classes smaller than 100.
  3. Twitter: $25\%, 25\%, 50\%$ for training, validation and test set.
     - Imbalanced ratio: 1:30.
- Baselines:
  - Over-sampling, Re-weight.
  - SMOTE, Embed-SMOTE.
  - GraphSMOTE$_T$, GraphSMOTE$_O$
  - GraphSMOTE$_{preT}$, GraphSMOTE$_{preO}$

Experiments

	Cora			BlogCatalog			Twitter		
Methods	ACC	AUC-ROC	F Score	ACC	AUC-ROC	F Score	ACC	AUC-ROC	F Score
Origin	0.681±0.001	0.914±0.002	0.684±0.003	0.210±0.004	0.586±0.002	0.074±0.002	0.967±0.004	0.577±0.003	0.494±0.001
over-sampling	0.692±0.009	0.918±0.005	0.666±0.008	0.203±0.004	0.599±0.003	0.077±0.001	0.913±0.006	0.601±0.011	0.513±0.003
Re-weight	0.697±0.008	0.928±0.005	0.684±0.004	0.206±0.005	0.587±0.003	0.075±0.003	0.915±0.005	0.603±0.004	0.515±0.002
SMOTE	0.696±0.011	0.920±0.008	0.673±0.003	0.205±0.004	0.595±0.003	0.077±0.001	0.914±0.005	0.604±0.007	0.514±0.002
Embed-SMOTE	0.683±0.007	0.913±0.002	0.673±0.002	0.205±0.003	0.588±0.002	0.076±0.001	0.943±0.004	0.606±0.005	0.514±0.002
GraphSMOTE$_T$	0.713±0.008	0.929±0.006	0.720±0.002	0.206±0.005	0.602±0.004	0.083±0.003	0.929±0.005	0.622±0.003	0.519±0.001
GraphSMOTE$_O$	0.709±0.010	0.927±0.011	0.712±0.003	0.215±0.010	0.591±0.012	0.080±0.005	0.905±0.008	0.616±0.006	0.515±0.003
GraphSMOTE$_{preT}$	0.727±0.003	0.931±0.002	0.726±0.001	**0.249**±0.002	**0.641**±0.001	**0.126**±0.001	0.937±0.003	**0.639**±0.002	0.531±0.001
GraphSMOTE$_{preO}$	**0.736**±0.001	**0.934**±0.002	**0.727**±0.001	0.243±0.002	**0.641**±0.002	0.123±0.001	**0.941**±0.002	0.636±0.001	**0.532**±0.001

Figure: Comparison of different approaches for imabalanced node classification.

Experiments

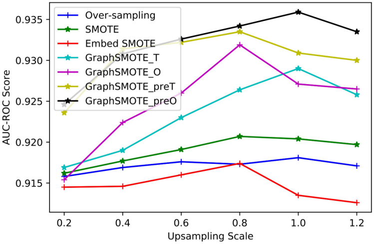


Figure: Affects of over-sampling scale on Cora dataset.

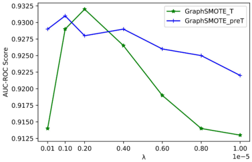


Figure: Affects of hyper-parameter λ on Cora dataset.

Experiments

	Imbalance Ratio			
Methods	0.1	0.2	0.4	0.6
Origin	0.8681	0.8998	0.9139	0.9146
over-sampling	0.8707	0.9039	0.9137	0.9215
Re-weight	0.8791	0.8881	0.9257	0.9306
SMOTE	0.8742	0.9027	0.9161	0.9237
Embed-SMOTE	0.8651	0.8967	0.9188	0.9212
GraphSMOTE$_T$	0.8824	**0.9162**	0.9262	0.9309
GraphSMOTE$_O$	0.8849	0.9061	0.9216	0.9311
GraphSMOTE$_{preT}$	**0.9167**	0.9130	0.9303	0.9317
GraphSMOTE$_{preO}$	0.9117	0.9116	**0.9389**	**0.9366**

Figure: Node classification performance on Cora under various imbalance ratios.

Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

William L Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1025–1035.

Nathalie Japkowicz. “The class imbalance problem: Significance and strategies”. In: *Proc. of the Int’l Conf. on Artificial Intelligence*. Vol. 56. Citeseer. 2000.

Hongyi Zhang et al. “mixup: Beyond empirical risk minimization”. In: *arXiv preprint arXiv:1710.09412* (2017).

Tianxiang Zhao, Xiang Zhang, and Suhang Wang. “GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks”. In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 833–841.